PARAMETRIC OPTIMIZATION OF ANALOG CIRCUITS USING NEURAL NETWORKS

A thesis submitted in

partial fulfillment of the requirements

for the degree of

Master of Technology

By

Arghajit Basu

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

April 2001.

CERTIFICATE

This is to certify that the work contained in the thesis entitled **"Parametric Optimization of Analog Circuits Using Neural Networks"**, by Mr. **Arghajit Basu** (Roll No. 9910415) has been done under my supervision and that this work has not been submitted elsewhere for the award of a degree.

19.4.01

(Dr. Aloke K. Dutta)

Associate Professor,
Department of Electrical Engineering,
Indian Institute of Technology Kanpur,
Kanpur - 208016, INDIA

April 2001.

ACKNOWLEDGEMENTS

On the completion of the work, I feel it as one of my greatest privileges to work under the noble and esteemed guidance of Dr. Aloke K. Dutta. I cannot forget him for his excellent supervision, skilled guidance and constant encouragement. His knowledge about my limitations, abilities and interests made me work with my full efficiency.

I express my sincere regards to Dr. S. Kar, Dr. B. Mazhari, Dr. A. Jain, and Dr. S. Roy whose courses have helped me to raise my knowledge level and understanding. I would like to thank Mr. Raju for providing excellent facilities at the Mayukh lab.

Thanks to all my classmates, kamlesh, rajarshi, santanu, kuldeep, prahallad, srikanth, bhanu, and krishna kumar for their warm support during my stay at IIT Kanpur. I also want to thank my friends, prithwi, saugata, arup, falguni, animesh who have made my stay a pleasurable and memorable one.

Words are not enough to express my feelings towards my parents and relatives. My parents always stood by my side to give me mental strength in my all endeavors. My acknowledgements will be incomplete if I do not mention about my uncle and aunt at Kanpur, who gave my stay a homely touch.

Arghajit Basu.

(Arghajit Basu)

ABSTRACT

Owing to the steady increase in the number of new application specific integrated circuit (ASIC) designs that include complex analog functions, the need for computer aided design (CAD) tools for analog circuits is being increasingly felt. In this work, we present a new design automation strategy that can fully automate the design path starting from the performance specifications and ending at a sized circuit schematic. This strategy relies on the radial basis function (RBF) network to predict the circuit performances from the design variables, and the penalty function method to solve a constrained optimization formulation of the synthesis problem of the circuit. In our method we do not need to formulate circuit equations in order to get the performance of the circuit from the design variables – another RBF network is also used in order to generate the initial values of the design variables from the given performance specifications. A database is created using SPICE for each topology, in order to train both the RBF networks. Using this approach, some of the basic circuit topologies in MOS technology are optimized, e.g., current sources (simple, cascode, and Wilson) and a common-source amplifier as a gain stage. The design routines for two basic CMOS op-amp topologies, e.g., the simple operational transconductance amplifier (OTA) and the basic compensated three-stage operational amplifier have been developed in this work. Once the optimization is done the program creates an output file to store the design variables, which can be used in SPICE simulation to compare the results with the given performance specifications. The design results obtained from our parametric optimization program of the circuit variables matched reasonably well with those obtained from SPICE simulation for each of the topologies designed in this work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

A surprising number of technologies that most people consider hallmarks of the digital revolution actually rely on *analog* circuitry core (e.g., op-amp); cellular telephones, magnetic disc drives, and compact disc players are just a few such examples. Many of tomorrow's products – e.g., neural networks, speech recognition systems, and personal digital assistants – will also require analog circuitry. Unfortunately, the present state of analog Computer-Aided Design (CAD) tools makes it difficult to quickly and cost effectively design the new analog circuitry that these new technologies require. To conserve space and save money, it is now commonplace to implement entire mixed analog/digital systems on a single Application Specific Integrated Circuit (ASIC). However, in order to maximize profit, mixed analog/digital ASIC designers must also minimize the design-time and thus the time-to-market. Digital CAD tools facilitate this by providing a rapid path to silicon for the large digital component of these designs.

Unfortunately, the analog component of these designs, although small in size, is still designed manually by experts using time-consuming techniques that have remained largely unchanged during the past twenty years or so. With the advent of logic synthesis tools and semicustom layout techniques to automate much of the digital design process, the analog

section may consume 90% of the overall design time, while consuming only 10% of the ASIC's die area [1]. Economic and other factors favor the co-existence of analog circuits, working either as the interface blocks or as the main signal processing circuits, and digital circuits on the same die. ASICs that are designed according to customer specifications, therefore, are moving steadily towards the integration of complete systems on a single-chip (SOC) [2].

The growing requirements for single chip mixed VLSI designs together with the pervasive trends towards smaller feature sizes and higher scales of integration have brought about new dimensions in circuit design complexity. Yet, while the design of digital circuits is thoroughly supported by sophisticated CAD tools, analog CAD tools lag considerably behind. In digital circuit design, hierarchy and structured abstractions are fully exploited in order to generate complex systems having large number of devices. In contrast, analog circuit design is mostly done manually by expert circuit designers. The techniques needed to build good analog circuits seem to exist solely as expertise invested within individual designers. CAD tools specifically tailored to analog integrated circuit design promise to improve the design process in a variety of ways as described below [3].

1) *By shortening the design time:* This will considerably improve productivity. Moreover, customer time-to-market will be satisfied easily.

2) *By simplifying the design process:* As a consequence, more designers, from novice to expert, will be able to design standard analog circuits.

3) *By improving the likelihood of error-free designs from the first fabrication run:* Automating error prone design tasks reduces the probability of making errors and therefore decreases the design cycle/success ratio.

4) *By reducing the design and production cost:* This will come about as a consequence of the shorter design times and smaller design cycles/success ratio.

5) *By improving the manufacturing yield:* Computer-aided methods can be used to estimate and enhance the manufacturing yield of circuits, thereby improving turnaround times and profitability.

6) *By allowing easier tracking of the fabrication processes:* CAD tools can be used to design circuits for different fabrication processes without much additional effort.

7) *By retaining the expert design knowledge:* The knowledge held by the design systems can be used over and over again to design new circuits. In addition, it can be conveyed to novice users in the form of design examples, explanations of behavior, suggestions for modifications or conclusions.

1.2 LITERATURE REVIEW

CAD of analog circuits has been slow in coming because the analog design process is far more complicated than the digital design. Whereas digital logic design needs only a small set of fixed building blocks, e.g., simple elements like AND gates, OR gates, and storage registers, or more complex blocks like arithmetic and logic units and memories; analog design, on the other hand, needs much more complex customized building blocks, ranging

from simple transistor pairs to multi-stage op-amps. A survey of the progress in the area of analog design automation (DA) can be found in the literature [4].

A CAD tool is a computer-based system that provides assistance to a design task. This assistance can range from merely relieving a designer from long, tedious and/or error-prone tasks of the design process to performing complete designs with minimal human intervention. This later class of CAD tools that automate part or whole of the design process are referred to as DA tools. Figure 1.2.1 depicts schematically a classification of the various design approaches.

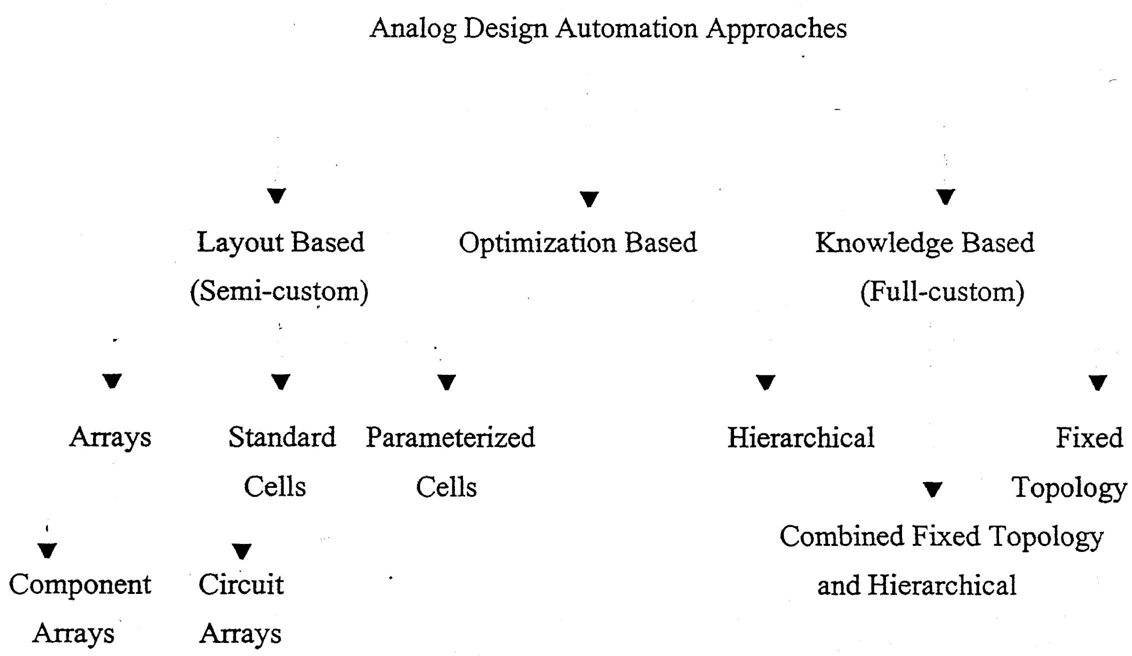

Fig. 1.2.1: A classification of analog IC design automation approaches [3].

A. Layout-Based Design Approach

With the sole aim to provide a fast and reliable path to silicon, several systems have been developed that follow a layout-based design approach. Thus, this approach is an adaptation of the extensively used standard cell, gate-array, and parameterized cell methods found in the digital domain. Since with this approach designs are controlled to an extent by the layout, it is also referred to as semi-custom bottom-up approach [3].

Analog arrays are pre-designed and laid-out blocks of different sizes, configurations, and levels of complexity, varying from single component arrays to circuit arrays. The required functions are designed by programming one or more levels of interconnect appropriately. Typical examples of circuit arrays are the SCA-6 and SCA-12 switched capacitor filter arrays with the assorted tools SLIDE and CAPSIZE developed by Silicon Systems, Inc. [5]. This method has several serious drawbacks.

i) They do not provide the necessary design flexibility required for high performance analog circuits, and restrict the designer to the limited range of available active and passive components, and also to a limited range of component values. In essence, analog arrays can only realize a small number of discrete points from a wide and continuous performance space.
ii) Arrays are not very cost-effective in terms of silicon usage, since any unused components or circuits in the array simply waste silicon area while providing no useful function.

Standard cells address this later problem of silicon usage. They are pre-designed and laid-out blocks of varying complexity that reside in the database of the DA system. The

required function is implemented by assembling the necessary cells and then by placement and routing. Only the required cells are fabricated at the end, hence, chip area is not wasted by unused blocks. Turnaround times, however, are longer than arrays and they too suffer from restricted design flexibility. It is also very difficult to configure and maintain a rich enough library of cells to accommodate such a wide spectrum of possible specifications.

Parameterized cells are similar to the standard cells, but with some additional flexibility gained by allowing some customization of the cells or parts of them according to the required function. The degree of flexibility provided is directly dependent on the sophistication of the respective module generator – the piece of code that generates the layout of the cell given a set of input parameters. In this way, components can now have a continuum of values – a significant feature not supported by the layout-based approaches described above. With the application of this approach, AIDE2 has demonstrated the design of several circuits including amplifiers, integrators, switched capacitor filters, and A/D converters.

B. Optimization-Based Design Approach

Historically, the first few attempts towards design automation were optimization-based. Systems such as DELIGHT.SPICE [6], ECSTACY [7], and ADOPT [8], consider the sizing of transistors of a user given circuit topology as an optimization problem. Typically, these systems employ optimization algorithms to iteratively adjust transistor sizes in order to meet user input constraints and objectives. A simulator is used within the optimization loop to assess the performance of the circuit during each iteration.

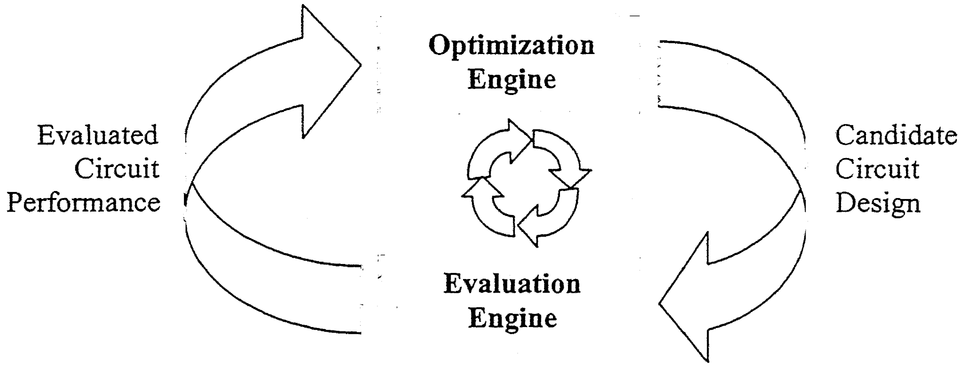

Fig. 1.2.2: Abstract model of analog synthesis tools [9].

Figure 1.2.2 illustrates the basic architecture of most analog synthesis tools. An *optimization engine* visits candidate circuit designs and adjusts their parameters in an attempt to satisfy designer-specified performance goals. An *evaluation engine* quantifies the quality of each circuit candidate for the optimizer. The optimizer is required to search the entire design space in order to arrive at the optimum design. Hence, it should visit as many circuit candidates as possible. However, the evaluator also has a speed limitation, i.e., there is a limit on the number of circuit candidates that it can evaluate, and, thus, there is a tradeoff between the optimizer and the evaluator. Even a small cell requires a mix of ac, dc, and transient analyses for its correct validation. It is perhaps not a surprise that analog synthesis strategies that rely on exotic, nonstandard, or fast-but-incomplete evaluation engines, have fared poorly in real design environments. To trust a synthesis result, one must first trust the methods used to quantify the performance of the circuit during synthesis [9]. Most prior works fail in this aspect.

C. **Knowledge-Based Design Approach**

Knowledge-based systems exploit domain knowledge to design analog integrated circuits, and they address the design task in a full-custom way, thereby allowing the maximum flexibility and a potentially better coverage of the circuit in the performance space.

- *Hierarchical Approach:* The underlying idea of hierarchical approach involves the breaking of the required circuit (or system) into smaller distinct parts or blocks. Each of these parts is assigned a set of specifications, and, if these are met, then the combined performance of all these parts will yield the desired circuit performance. The same procedure is repeated in a similar manner for the smaller blocks at lower hierarchical levels. To be able to carry out such partitioning of circuits and decompositions of specifications, a great deal of domain knowledge is required, which is mainly in the form of design equations and heuristics. Different variants of the basic hierarchical design approach have been realized by a number of systems, such as BLADES [10], OASYS [11], and An_Com [12].
- *Fixed Topology Approach:* The fixed topology approach employs a sizing method to compute appropriate sizes for the devices within a fixed circuit topology. These fixed, unsized, device level circuit topologies are stored in a knowledge base, together with the necessary domain knowledge for dimensioning the devices. Some of the systems that follow this approach include IDAC [13], OPASYN [14], and OAC [15].
- *Combined Hierarchical and Fixed Topology Approach:* Some other knowledge-based design methods combine the features of both the hierarchical and fixed topology approaches. ASAIC [16] is a system that fits into this class of design systems, since it

puts together a circuit topology in a hierarchical manner whereas the design (i.e., sizing) of the topology is performed in a manner that resembles fixed topology systems.

1.3 OBJECTIVE OF THE PRESENT WORK

Previous approaches to analog circuit synthesis have failed to make the transition from research to practice. This is primarily due to the prohibitive one-time effort required to derive the complex equations that run these synthesis tools. Because they rely on a set of equations, the previous approaches to synthesis discussed earlier are referred to as equation-based, and their architecture is discussed in terms of the simplified search loop, as shown in Fig.1.3.1.

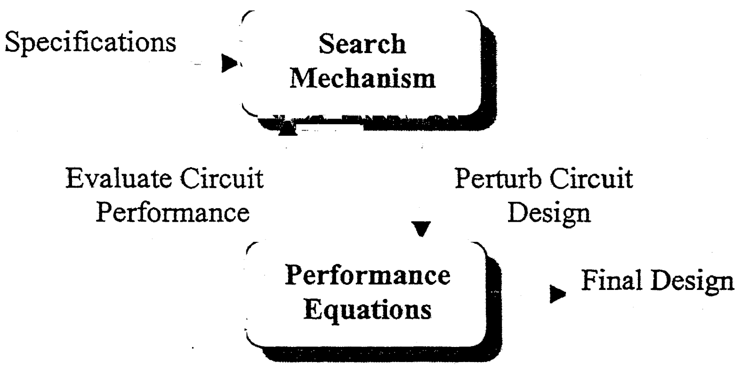

Fig. 1.3.1: Search process used in the equation-based analog synthesis tools [1].

At each step in the synthesis process (i.e., during each pass through this loop), a search mechanism perturbs the element values, transistor dimensions, and other variable aspects of the circuit in an attempt to arrive at a design that more closely meets the

performance specifications and other objectives. Performance equations are used to evaluate the new circuit design, determine how well it meets the specifications, and provide feedback to the search mechanism to guide its progress.

Equation-based approaches rely heavily on simplifications to circuit equations and device models. Consequently, the performance of the synthesized circuit often reflects the limitations of the simplified equations used to model it, rather than the inherent limitations of the circuit topology or the underlying fabrication technology. The need for designs that push the limits of circuit topologies and use the latest technologies invalidates the use of many of these simplifications. For example, for a MOS transistor in saturation, the drain current I_{DS} is given by $I_{DS} = \frac{K'W}{2L}(V_{GS} - V_T)^2$, where K' is the process transconductance parameter, W is the channel width, L is the channel length, V_{GS} is the gate to source voltage, and V_T is the threshold voltage of the transistor. This is a workable model of the current-voltage relationship for a device fabricated by a 3 μm process. However, this can yield grossly inaccurate performance predictions for a device with sub-micron channel lengths [1].

Equation-based tools appear to design circuits quickly. However, the run-times of these tools are not an accurate measure of automation because they do not consider the preparatory time required to derive the circuit equations. Even for a relatively simple analog circuit, these equations are very complex, require considerable analog design expertise to derive, and must be entered as thousands of lines of program code. For a textbook design, this process can take weeks, while for an industrial design it can take several designer-years [14]. Moreover, adding these equations typically requires a user who is a programmer, an analog

designer, and an expert intimate with the internal architecture of the tool, all at the same time. As a result, it is almost always easier for an industrial mixed-signal ASIC designer to design circuits manually rather than dedicate the effort required to teach these tools to do it.

Among the various sub-domains of analog design, i.e., topology selection, parametric optimization, and layout generation, each one plays an important role to achieve higher performance for a particular application. Even if the topology of a circuit to be designed is fixed, finding the optimal set of circuit parameter values is not an easy and obvious task, since all the circuit performances are highly dependent on these choices. The parametric optimization is complicated by conflicting design objectives and performance constraints, which are generally implicit nonlinear functions of the circuit parameters.

In this work, an analog CAD tool is developed in order to optimize the values of the design variables for a given set of performance specifications, where the topology of the circuit to be designed is known a priori. Since this is an optimization-based approach, hence, we need to evaluate the circuit performances for different sets of design parameters. In this work, this operation is done by a radial basis function (RBF) network, which is trained previously by a database created using SPICE. On the other hand, equation-based DA tools achieve this by modeling simple circuit equations, which is not an easy task for complex circuits. However, we do not have to formulate the circuit equations to obtain the performance of the circuit from the design parameters, since this is obtained by using the RBF network. Hence, the time-consuming circuit simulator is avoided in the optimization loop. The optimization problem is carried out using the penalty function method, which is a constrained optimization technique. In this method, the constrained problem is transformed

into a sequence of unconstrained problems by adding penalty terms to the objective function for each constraint violation. The approach adopted in our work for the synthesis of analog circuits combines the advantages of both the optimization-based and the knowledge-based approaches and tries to overcome the limitations of both of them. This approach has the following advantages over the other DA approaches discussed earlier.

1. The training data for the RBF network is acquired using SPICE, and, hence, the circuit realization is more accurate than the other methods which used simple model equations. However, the accuracy of the results is directly related to the level of the SPICE model, which is used to from the training data.
2. The penalty function method is a constrained objective algorithm, which can take care of the constrained design specifications.
3. The initial values of the design parameters are also obtained from a trained RBF network, and, thus, the optimization loop requires less number of iterations.
4. The RBF network is used to avoid a simulator in the optimization loop, and, therefore, the cost of repeated simulation is avoided.

In order to explore the viability of the analog circuit design methodology as stated above, a CAD tool is constructed in this work in the C-language and applied for the synthesis of analog blocks using MOS technology. The most widely used analog building block is the op-amp, which is almost an indispensable component of all analog systems. Synthesis of op-amps shows the nature of intricacies involved in the design of analog circuits; performances of op-amps track in different directions, and, therefore, optimization proves to be a challenging task for analog circuit CAD tool developers. Realizing the immense importance

of op-amps in the designs of analog modules, we have applied the proposed approach for optimization for two different op-amp topologies (i.e., the simple operational transconductance amplifier (OTA) and the compensated basic three-stage op-amp). Also, as a first step in order to validate our approach, the parametric designs of a variety of current sources and a common-source amplifier as a gain stage have been automated in this work. The details of the performance specifications and the synthesis procedure for the above-mentioned analog building blocks are described in the subsequent chapters. The results obtained from the parametric optimization of different topologies are discussed along with the design routines in the respective chapters.

The development of the synthesis tool involved the creation of an optimization module in order to optimize an n-variable constrained objective function. In order to develop the optimization modules for each of the topologies currently supported by our program, the following steps are followed.

- Identify the performance specifications and design variables of a circuit topology.

- Eliminate the unimportant and dependent design variables by using heuristics and circuit knowledge, and, thus, filter out the independent design variables.

- For each circuit, prepare the database to train the RBF networks. After training, the parameters of the network (e.g. weights, etc.) are stored in a file.

- Develop the program to read the performance specifications and to create output files for storing the output results in a result file.

- Verify the module by running it for a number of sets of performance specifications, obtain the output, and compare the results with those obtained from SPICE simulation.

The results as obtained from our program for all the modules match reasonably closely with those obtained from the SPICE simulations. The methodology adopted in this work can be interpreted as the development of a part of a complete analog circuit design automation tool. In other words, our attempt is to provide reliable and useful parametric optimization procedures for the synthesis of cell level analog circuit blocks. The tool developed here can be used for evaluating the circuit parameters, considering the important performances for the design of the circuit. After the design, SPICE simulation can be used to verify the circuit performances with the given specifications. The approximate values of the specifications that can be achieved with a circuit should be known to the user in order to avoid the possibility of providing absurd values as performance specifications. Otherwise, the tool may not produce an optimal solution, which is a problem for most of the optimization algorithms.

The reminder of this thesis is organized in the following manner. Chapter 2 describes the design automation procedure in detail. It explains the formulation of the radial basis function networks and focuses on the penalty function method, which is used as an optimization routine. Chapter 3 presents the design results of some simple MOS circuits (e.g., current mirrors and a common source gain stage). This DA procedure is applied for the parametric design of two types of op-amp (i.e., simple OTA and compensated three stage), and the results are given in Chapter 4. Finally, Chapter 5 presents the concluding remarks.

CHAPTER 2

THE DESIGN AUTOMATION (DA) PROCEDURE

2.1 INTRODUCTION

This chapter deals with the explanation of the prototype DA program that has been developed in this work, and is applied for the synthesis of cell level analog circuits. The general structure of the synthesis procedure adopted in this work can be represented by the means of a flow chart as shown in Fig.2.1.1. The circuit selection and the layout generation tasks for the circuits are not implemented in this work, and, therefore, these are shown in dotted lines in Fig.2.1.1.

The explanation of the synthesis algorithm goes as follows. First of all, a database has to be created in order to train the radial basis function (RBF) network for the circuit to be designed. This database is developed using SPICE, and is created only once for a particular circuit. This database is used to train two networks: one is for providing the values of the initial design parameters from the user given performance specifications (this network is termed RBF1 in Fig.2.1.1), and the other one is for calculating the performance from the circuit description in the optimization module (termed RBF2 in Fig.2.1.1). Now, both RBF networks for that circuit are trained (i.e., the network is tuned to the circuit parameters) and stored (i.e., the parameters of the network, like weights, etc., are saved). The main program starts with a set of input performance specifications (e.g., for an op-amp, these could be the open loop gain, phase margin, slew rate, area, etc.), and deduces the initial set of design

parameter values (e.g., compensation capacitors, length and width of transistors, bias currents, etc.) by RBF1.

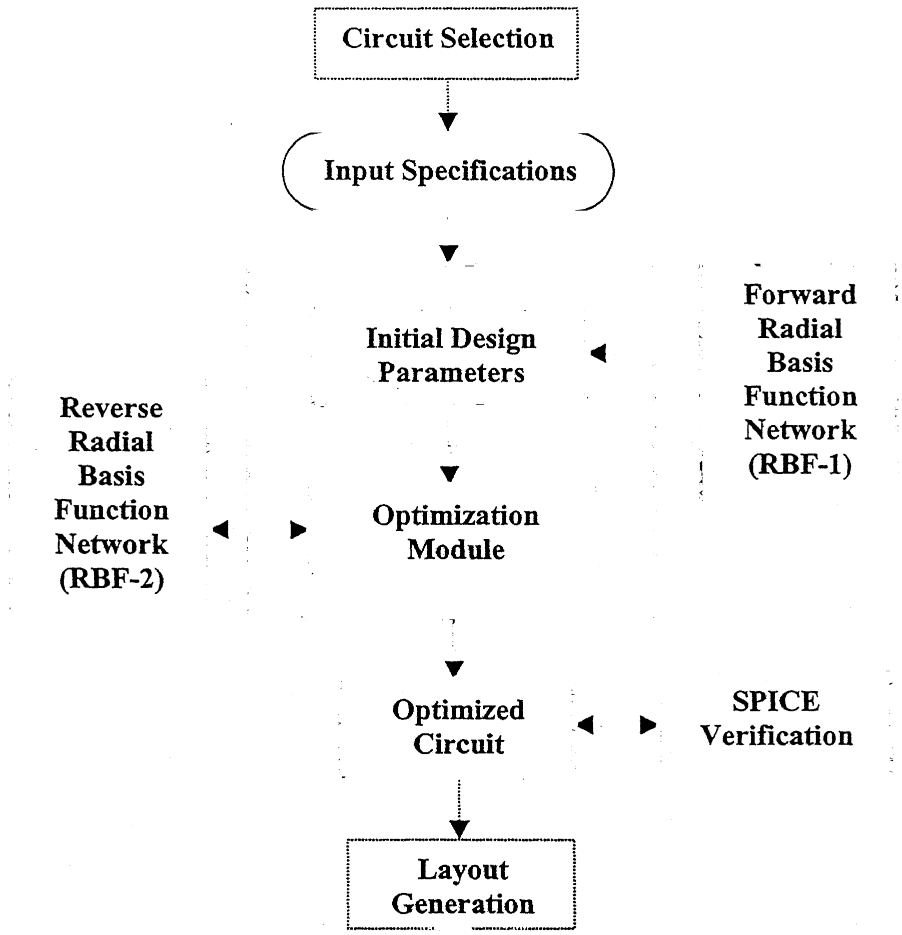

Fig. 2.1.1: The general structure of the synthesis algorithm.

The optimization module starts with the initial solution and after several iterations identifies the optimal set of parameter values with the help of RBF2. In this work, RBF2 is used to get the circuit performance from the design parameters. Actually, RBF2 replaces the

circuit equations that used to model the circuit to be designed. The parameter values obtained satisfy all the design objectives and constraints. However, if no solution exists for the given set of specifications, then the optimization module gives the result, which is the best among a set of results for that particular case. The program is aimed at arriving at a solution in the design space in order to obtain the designed values of the performances close to their corresponding specified values, which may not be fully met. The output results, i.e., the design variables and the performances of the circuit are stored in a file. This can be used to compare the results obtained from the SPICE simulation.

2.2 THE OPTIMIZATION MODULE

In order to apply the optimization algorithm discussed earlier for design of circuits, a mapping of the design problem into a mathematical function is required, which can be solved using the optimization technique. Unconstrained optimization is not suitable for synthesis of ICs, since there are always some constraints on the performance specifications and the design parameters. Hence, constrained optimization approach is adopted in this work.

A constrained optimization problem comprises of an objective function together with a number of equality and inequality constraints. Often lower and upper bounds on design variables are also specified. A constrained problem is written as follows:

Minimize $f(x)$,

subject to:

$$g_i(x) > or = 0, \qquad i = 1,2,\ldots,n,$$

$$\text{and } x_{\min} \le x \le x_{\max}, \tag{2.1}$$

where $f(x)$ is the objective function which is to be minimized, $g_i(x) > or = 0$ are n equality or inequality constraints which are to be satisfied, x is the vector of the design parameters (e.g., lengths and widths of various transistors, values of compensation capacitor, bias voltages and currents, etc.), and x_{min} and x_{max} are the minimum and the maximum values of the design variables respectively.

The constrained problem is transformed into a sequence of unconstrained problems by adding penalty terms for each constraint violation, i.e., if a constraint is violated at any point, the objective function is penalized by an amount depending on the extent of constraint violation. In this work, the penalty function method [17] is used in order to optimize the design variables. This method works in a series of sequences, each time modifying a set of penalty parameters and starting a sequence with the solution obtained in the previous sequence. At any sequence, the following penalty function is minimized [17]:

$$P(x,R) = f(x) + \Omega(R, g_i(x)), \tag{2.2}$$

where R is a set of penalty parameters, Ω is the penalty term chosen to favour the selection of feasible points over infeasible points. For equality or inequality constraints, different penalty terms are used, as discussed below [17].

♦ *Parabolic penalty:* $\Omega = R\{g_i(x)\}^2$.

This penalty term is used for handling equality constraints. Since the feasible point satisfies $g_i(x) = 0$, any infeasible point is penalized by an amount proportional to the square of the constraint violation. The parameter R controls the extent of this penalty, i.e., if R is large then the penalty term added due to a constrained violation is also large.

♦ *Infinite barrier penalty:* $\Omega = R\sum_{j \in \bar{J}} |g_i(x)|$.

This penalty term is also used for handling equality constraints; hence, if $|g_i(x)| = 0$, then, no penalty term is added, but if $|g_i(x)| \neq 0$ then, a penalty proportional to the constraint violation is added to the objective function. Here, the term R is a large number (usually ~ 10^{20}) in order to get large penalties for infeasible points, and $\bar{J}$ denotes the set of violated constraints at the current point.

♦ *Log penalty:* $\Omega = -R\ln[g_i(x)]$.

This penalty term is used for inequality constraints. For infeasible points, $g_i(x) < 0$, thus, this penalty term cannot assign penalty to infeasible points. For feasible points, more penalty is assigned to points close to the constraint boundary or points with very small $g_i(x)$. As an example, consider the case of the open loop gain of an op-amp; the point, which is much larger than the specified value, is more desirable than the point closer to that. Hence, the points closer to the boundary are penalized more in order to get better results. As an example, for $R = 10$, $\Omega = 115.1$ when $g_i(x) = 10^{-5}$, and $\Omega = 230.3$ when $g_i(x) = 10^{-10}$.

Thus, the points closer to the boundary $g_i(x) = 0$, are penalized more than the points further away from that.

- *Inverse penalty:* $\Omega = R\left[\frac{1}{g_i(x)}\right]$.

Similar to the log penalty term, this term is also suitable for inequality constraints. This term penalizes only feasible points – the penalty is more for boundary points. Again, from the example given above, for $R = 10$, $\Omega = 10^6$ when $g_i(x) = 10^{-5}$, and $\Omega = 10^{11}$ when $g_i(x) = 10^{-10}$.

- *Bracket operator penalty:* $\Omega = R\langle g_i(x)\rangle^2$,

where $\langle \alpha \rangle = \alpha$ when α is negative, and zero otherwise. This term also handles inequality constraints. The bracket operator assigns positive penalty values to the infeasible points.

The penalty term *R* needs to be modified because a small initial value of *R* results in an optimum solution close to the unconstrained optimum point (as small penalty is added to the penalty function for a constrained violation due to small *R*). As *R* is increased in successive sequences, the solution improves and finally approaches the true constrained optimum point.

The penalty function algorithm is described with an example as follows [17].

Step 1: Choose two termination parameters ε_1 and ε_2, where ε_1 is used to terminate the unconstrained search and ε_2 is the minimum difference in the penalized function values at

two consecutive sequences. Also, choose an initial solution $x^{(0)}$, a penalty term Ω, and an initial parameter $R^{(0)}$. Choose a parameter c, which is used to update R (as in Step 5) and set $t = 0$, where t is the iteration number.

Step 2: Form $P(x^{(t)}, R^{(t)}) = f(x^{(t)}) + \Omega(R^{(t)}, g_i(x^{(t)}))$ [refer to Eq.(2.2)].

Step 3: Starting with a solution $x^{(t)}$, find $x^{(t+1)}$, such that $P(x^{(t+1)}, R^{(t)})$ is a minimum for a fixed value of $R^{(t)}$. Use ε_1 to terminate the unconstrained search.

Step 4: Is $\left| P\left(x^{(t+1)}, R^{(t)}\right) - P\left(x^{(t)}, R^{(t-1)}\right) \right| \leq \varepsilon_2$?

If yes, set $x^{(t)} = x^{(t+1)}$ and terminate;

Else go to *Step 5*.

Step 5: Choose $R^{(t+1)} = cR^{(t)}$. Set $t = t + 1$ and go to *Step 2*.

Example:

Consider the constrained function:

$$\text{Minimize} \quad \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

subject to

$$\left(x_1 - 5\right)^2 + x_2^2 - 26 \geq 0, \qquad x_1, x_2 \geq 0 .$$

Step 1: We use the bracket operator penalty term to solve this problem. We choose an infeasible point $x^{(0)} = (0,0)$ as the initial guess. We also choose a small value for the penalty parameter, $R^{(0)} = 0.1$, $c = 10$, and two convergence parameters $\varepsilon_1 = \varepsilon_2 = 10^{-5}$.

Step 2: The next task is to form the penalty function:

$$P\left(x, R^{(0)}\right) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2 + 0.1 \times \left\langle \left(x_1 - 5\right)^2 + x_2^2 - 26 \right\rangle^2 .$$

Step 3: The above unconstrained function can now be minimized using the steepest descent method [17]. We begin the algorithm with an initial solution $x^{(0)} = (0,0)$ having $f(x^{(0)}) = 170.0$. At this point, the constraint violation is -1.0 and the penalized function value $P(x^{(0)}, R^{(0)}) = 170.10$. Using the unconstrained search, after about 150 function evaluations, the solution $x^{*} = (2.628, 2.475)$ having a function value equal to $f(x^{*}) = 5.695$ is obtained. At this point, the constraint violation is equal to –14.248, but has a penalized function value equal to 25.996, which is smaller than that at the initial point. Even though the constraint violation at this point is greater than that at the initial point, the steepest descent method has minimized the penalized function $P(x, R^{(0)})$ from 170.10 to 25.996. We set $x^{(1)} = (2.628, 2.475)$ and proceed to the next step.

Step 4: Since this is the first iteration, we have no previous penalized function value to compare with; thus, we move to Step 5.

Step 5: At this step, we update the penalty parameter $R^{(1)} = 10 \times 0.1 = 1.0$ and move to Step 2. This is the end of the first sequence.

Step 2: The new penalized function in the second sequence is as follows:

$$P(x, R^{(1)}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + 1.0 \times \langle (x_1 - 5)^2 + x_2^2 - 26 \rangle^2 .$$

Step 3: At this step, we once again use the steepest descent method to solve the above problem from the starting point (2.628, 2.475). The minimum of the function is found after about 340 function evaluations and is $x^{(2)} = (1.011, 2.939)$. At this point, the constraint violation is equal to –1.450, which suggests that the point is still an infeasible point.

Step 4: Comparing the penalized function values, we observe that $P(x^{(2)}, 1.0) = 58.664$ and $P(x^{(1)}, 0.1) = 25.996$. Since they are very different from each other, we continue with Step 5.

Step 5: The new value of the penalty parameter is $R^{(2)} = 10$. We increment the iteration counter to $t = 2$ and proceed to Step 2.

In the next sequence, the penalized function is formed with $R^{(2)} = 10$. This time the steepest descent algorithm starts with an initial solution $x^{(2)}$. The minimum point of the sequence is found to be $x^{(3)} = (0.844, 2.934)$ with a constraint violation equal to –0.119. After another sequence (iteration) of this algorithm, the obtained solution is $x^{(4)} = (0.836, 2.940)$ with a constraint violation of only –0.0175. This point is very close to the true constrained optimum solution. A few more iterations of the penalty function method may be performed in order to obtain a solution with the desired accuracy.

The main advantage of this method is that any type of constraint can be handled. The algorithm does not take into account the structure of the constraints, i.e., both linear and nonlinear constraints can be handled easily with this algorithm. However, in the presence of multiple constraints, the transformation as given in Eqn.(2.2) can cause the penalized function to have artificial local optima, where the penalty function algorithm may get attracted to. The performance of the penalty function can be improved in presence of this problem, if the constraints and the objective function are first normalized before constructing the penalized function. As an example, inequality constraint $g_i(x) \geq 0$ can be normalized as follows:

$$0 \leq \frac{g_i(x)}{g_{\max}} \leq 1,$$

where g_{max} is the maximum value of the constraint $g_i(x)$ in the search space. If an upper bound of the objective function is known, the objective function can also be normalized as shown above [17].

2.3 THE RADIAL BASIS FUNCTION (RBF) NETWORK

The design of a supervised neural network (the network which uses input and output data for its training) may be pursued in a variety of different ways. The design of a neural network can be viewed as a *curve-fitting* (approximation) problem in a high-dimensional space (where the number of independent variables is high). According to this viewpoint, learning is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data, with the criterion for "best fit" being measured in some statistical sense. Correspondingly, generalization is equivalent to the use of this multidimensional surface (which is created by the network after training) to interpolate the test data. In the context of a neural network, the hidden units (or layers) provide a set of "functions" that constitute an arbitrary "basis" for the input patterns (or vectors) when they are expanded into the hidden-space; these functions are called the *radial-basis functions* (RBF).

The construction of a RBF network in its most basic form involves three entirely different layers. The input layer is made up of source nodes (also referred to as the sensory units). The second layer is a hidden layer of high enough dimension (i.e., the number of nodes in the hidden layer may be large). The output layer supplies the response of this

network to the activation patterns applied to the input layer. The transformation from the input space to the hidden-unit space is nonlinear, whereas the transformation from the hidden-unit space to the output space is linear.

A schematic of the RBF network with N inputs and a scalar output is depicted in Fig.2.3.1. Although the scalar output case is considered here for notational simplicity, extension to the multi-output (i.e., vector output) case is straightforward. Since the input to the hidden layer transformation is same in the multi-output case, only the number of weight vectors from the hidden layer to the output layer is increased. In fact, a multi-output RBF network can always be separated into a group of single output RBF networks [18].

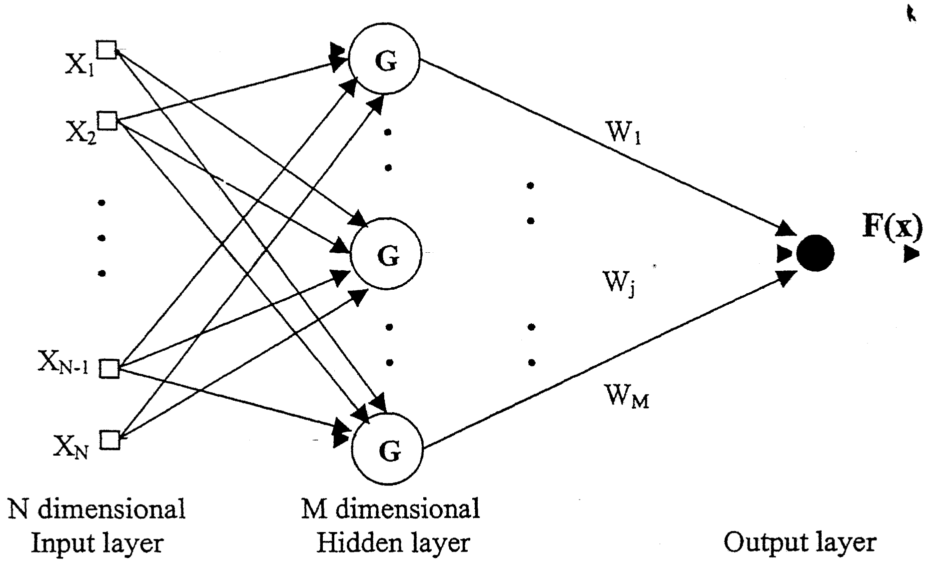

Fig. 2.3.1: A radial basis function network with N inputs, M hidden layers (with $M \leq N$), and a scalar output F [18].

This type of network implements a mapping $f_r : \mathbf{R}^N \rightarrow \mathbf{R}$ according to

$$f_r(x) = \sum_{i=1}^{M} w_i G(\|x - c_i\|), \tag{2.3}$$

where $x \in R^N$ is the input vector, $G(\cdot)$ is a given function from $\mathbf{R}^N$ to $\mathbf{R}$, $\|\cdot\|$ denotes the Euclidean distance (i.e., the sum of difference squares), w_i $(1 \leq i \leq M)$ are the weights of the parameters to calculate the output from the hidden layer, $c_i \in R^N, 1 \leq i \leq M$, are known as the RBF centers, and M is the number of such centers. In an RBF network, the functional from $G(\cdot)$ and the centers c_i are assumed to be fixed. By providing a set of the input $x(t)$ and the corresponding desired output $d(t)$ for $t = 1$ to N, the values of the weights w_i can be determined by a method described later in this chapter. However, the choices of $G(\cdot)$ and c_i must be carefully considered in order for the RBF network to be able to match closely the performance of the two layer neural network.

Theoretical and practical results suggest that the choice of the nonlinearity $G(\cdot)$ is not crucial to the performance of the RBF network [19]. For example, the thin-plate-spline-function [19]

$$G(v) = v^2 \log(v), \tag{2.4}$$

and the Gaussian function

$$G(v) = \exp\left(-v^2 / \beta^2\right), \tag{2.5}$$

where β is a real constant, are two typical choices. For the nonlinearity given by Eqn.(2.4), $G(v) \rightarrow \infty$ as $v \rightarrow \infty$, and for Eqn.(2.5), $G(v) \rightarrow 0$ as $v \rightarrow \infty$. Although these two

nonlinearities have quite different properties, both the resulting RBF networks have good approximation capabilities, i.e., it can interpolate the test data adequately. In practice, the centers are normally chosen from the data points. Typically, the number of hidden nodes is less than the number of data points (i.e., $M \leq N$).

A Gaussian RBF of a vector x with the center at c_i may be expressed as [18]

$$G(\|x-c_i\|)=\exp\left\{-\sigma(x-c_i)^T\Sigma^{-1}(x-c_i)\right\}, \tag{2.6}$$

where σ determines the standard deviation of the function, and Σ is the covariance matrix of the input vectors and is defined as

$$\Sigma_{ij}=\sum_{k=1}^{N}\left(x_{ki}-\overline{x_i}\right)\left(x_{kj}-\overline{x_j}\right), \tag{2.7}$$

where $\overline{x}$ is the mean of the same type of data (e.g., the gain of an op-amp).

This particular choice of the basis function is the only one that guarantees that in the case of $M = N$, and $c_i = x_i$, for $i = 1, 2, \ldots\ldots, N$, the correct solution of Eqn.(2.3) is consistently recovered, or, in other words, the mapping is done correctly.

The problem one has to address is the determination of the new set of weights $\{w_i \mid i = 1, 2, \ldots, M\}$ so as to minimize the new cost function $\mathscr{E}(F^*)$, which may also be termed as the error function and is defined by [18]

$$\mathscr{E}(F^*)=\sum_{i=1}^{N}\left(d_i-\sum_{j=1}^{M}w_jG(\|x-c_i\|)\right)^2. \tag{2.8}$$

The term on the right hand side of Eqn.(2.8) is similar to the squared Euclidean norm $\|d - Gw\|^2$, where

$$d = [d_1, d_2, ..., d_N]^T,$$

$$G = \begin{bmatrix} G(x_1;t_1) & G(x_1;t_2) & \cdot & \cdot & \cdot & G(x_1;t_M) \\ G(x_2;t_1) & G(x_2;t_2) & \cdot & \cdot & \cdot & G(x_2;t_M) \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ G(x_N;t_1) & G(x_N;t_2) & \cdot & \cdot & \cdot & G(x_N;t_M) \end{bmatrix}, \tag{2.9}$$

and $w = [w_1, w_2, ..., w_M]^T$.

The desired response vector d is N dimensional as before. However, the matrix G and the weight vector w have different dimensions; the matrix G is N-by-M, and the vector w is M-by-1. The minimization of Eq.(2.8) with respect to the weight vector w yields the result

$$(G^T G)w = G^T d. \tag{2.10}$$

Hence, the weight vector w converges to the pseudo-inverse (minimum-norm) solution for the problem, given by

$$w = G^+ d, \tag{2.11}$$

where G^+ is the pseudo-inverse of matrix G; that is,

$$G^+ = (G^T G)^{-1} G^T. \tag{2.12}$$

Now, the new output is calculated by

$$o_i = \sum_{j=1}^{M} w_{ij} G(\|x - c_i\|), \quad i = 1,2,...N. \tag{2.13}$$

Then, the weights are updated using the equation

$$\frac{\partial\Phi(n)}{\partial w_j(n)} = 2\eta\sum_{i=1}^{N} G(\|x - c_i\|)(o_i - d_i)^2, \tag{2.14}$$

and $$w_j(n+1) = w_j(n) - \frac{\partial\Phi(n)}{\partial w_j(n)}, \quad j = 1,2,\ldots,M, \tag{2.15}$$

where n is the iteration number and η is the learning rate, which is used to update the weight vectors. Also, $\partial\Phi/\partial w_i$ is the rate of change of the weight in each iteration while minimizing the error function. Thus, the network is trained until a previously specified error goal is achieved or the iteration number crosses the maximum limit. Finally, when either of these two conditions gets satisfied, the network becomes trained and the weights and the centers are stored in a file for future use.

CHAPTER 3

DESIGN OF SIMPLE MOS CIRCUITS

3.1 INTRODUCTION

In this chapter, a few design examples are presented in order to illustrate the circuit design automation procedure developed in this work here and discussed earlier in Chapter 2. Simple MOS current mirrors and a common source gain stage are used as the design examples. The technology parameters (e.g., threshold voltage, oxide thickness, channel length modulation parameter, etc.), which are used for SPICE simulation, are taken for a 0.8 μm process [20]. The maximum length (L) of the transistor is taken to be 100 μm, and the minimum feature size is taken to be 0.8 μm. The circuit schematic along with its performance specifications are presented for each circuit. The design results, as obtained from the program, for a set of performance specifications for each of the topologies are also presented. These results are used for the SPICE simulations, the output of which are then compared with the given performance specifications.

3.2 CURRENT MIRRORS

Design of current sources using active devices have come to be widely used in analog integrated circuits both as biasing elements and as load devices for amplifier stages. The use of current sources in biasing can result in superior insensitivity of circuit performance to power-supply variations and to temperature. Current sources are more economical than

resistors in terms of the die area required to provide bias current of a certain value, particularly when the required value of the bias current is small. When used as a load element in transistor amplifiers, the high incremental resistance of the current source results in a high voltage gain at low power supply voltages.

The following different types of current mirrors are designed and simulated in this work.

1. Simple current mirror.
2. Wilson current mirror.
3. Cascode current mirror.

Depending on the requirements, a particular current mirror is chosen keeping in mind the performance it can provide. For the design of current mirrors, the designer specifies the performance specifications, e.g., the total active area, output current, output impedance, and minimum output voltage. The design variables are the width (W) and the length (L) of the transistors. In order to reduce the total number of independent variables, all active transistors in a particular current source are chosen to have the same size, i.e., they have the same lengths and the same widths. Thus, the bias current I_{bias} is equal to the output current I_{out} for all the cases considered here.

In this work, the bias current (I_{bias}) source is implemented using a depletion-mode NMOS transistor for all current sources. Thus, the length (L_0) and the width (W_0) of the load transistor are considered to be design variables. Body effect of the transistors is also considered for all the topologies.

3.2.1 Simple Current Mirror

The schematic of a simple current mirror is shown in Fig.3.2.1 along with the node numbers used in SPICE simulation. The sources of all the transistors are connected to V_{SS}.

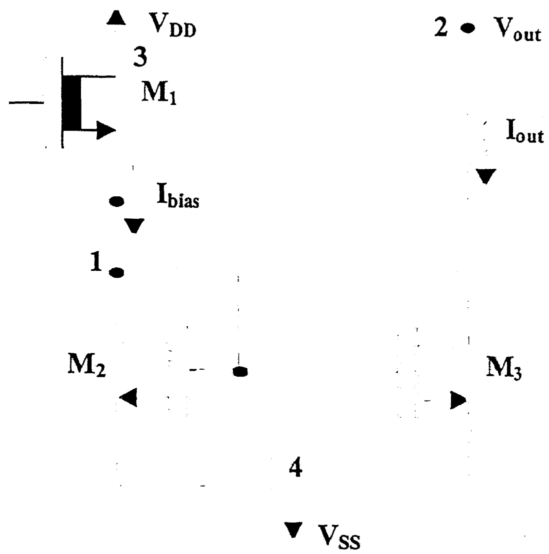

Fig. 3.2.1: Schematic of a simple current mirror.

The transistors M_2 and M_3 are taken to be of the same size. Hence, there are four design variables, namely, widths and lengths of M_1 and M_2. Based on this assumption, a training database for the simple current mirror is created using SPICE simulation. This database is used to train both the forward (RBF1 in Fig.2.1.1) and the reverse (RBF2 in Fig.2.1.1) radial-basis function networks, as discussed in Chapter 2.

Based on the above training data, a module has been developed for the simple current mirror, which gives the design variables as output from a given set of performance specifications as input to the routine. The output results along with the results obtained from

SPICE simulation, and the design variables obtained for a given set of performance specifications are shown in Tables 3.2.1.1 and 3.2.1.2 respectively. The results obtained from our algorithm match well with those obtained from SPICE simulation.

Table 3.2.1.1

The design results obtained for a simple current mirror and their comparison with those obtained from SPICE simulation.

Sl. No.	Performance	Type	Specified value	Value obtained from program	Value obtained from SPICE
1.	I_{out} (μA)	=	15	15.377	15.48
2.	R_{out} (MΩ)	≥	1	1.671	1.676
3.	V_{outmin} (V)	≤	0.3	0.229	0.239
4.	$AREA$ (μ^2)	≤	20	19.044	-

Table 3.2.1.2

The design variables obtained from the optimization program for a simple current mirror.

Sl. No.	Design variable	Value
1.	Width/length of M_2, M_3 (μm/μm)	9.929/0.8
2.	Width/length of load transistor M_1 (μm/μm)	0.8/3.947

3.2.2 Wilson Current Source

A widely used current source that achieves very high output resistance is the Wilson configuration. Figure 3.2.2 shows the schematic of a Wilson current source considered in this work along with the node numbers used in SPICE simulation. All the bodies of the transistors are connected to V_{SS}. This circuit actually uses a negative feedback with feedback through M_3 activating M_2 to raise the output resistance.

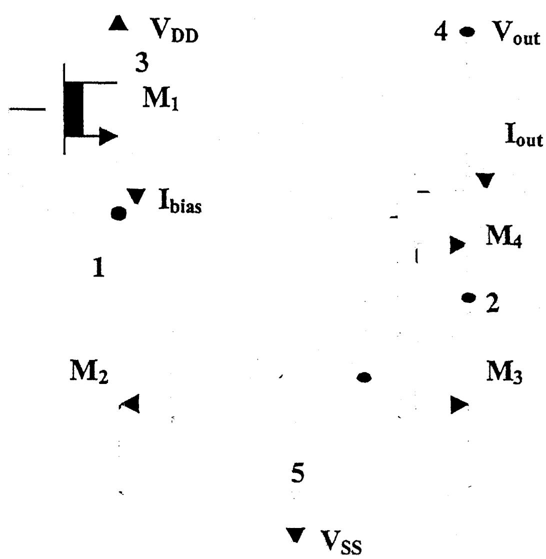

Fig. 3.2.2: Schematic of a Wilson current source.

The transistors M_2, M_3, and M_4 are taken to be of the same size. Hence, there are four design variables, such as, widths and lengths of M_1 and M_2. Based on this assumption, a training database for the Wilson current source is created using SPICE simulation. This database is used to train both the RBF networks.

Based on the above training data, a module has been developed for the Wilson current source, which gives the design variables as output from a given set of performance specifications as input to the routine. The output results along with the results obtained from SPICE simulation, and the design parameters obtained for a given set of performance specifications are shown in Tables 3.2.2.1 and 3.2.2.2 respectively. The results obtained from our algorithm match well with those obtained from SPICE simulation.

Table 3.2.2.1

The design results obtained for a Wilson current source and their comparison with those obtained from SPICE simulation.

Sl. No.	Performance	Type	Specified value	Value obtained from program	Value obtained from SPICE
1.	I_{out} (μA)	=	30	30.92	32.5
2.	R_{out} (MΩ)	≥	15	15.47	17.32
3.	V_{outmin} (V)	≤	1.5	1.31	1.27
4.	$AREA$ (μ^2)	≤	100	72.5	-

Table 3.2.2.2

The design variables obtained from the optimization program for a Wilson current source.

Sl. No.	Design variable	Value
1.	Width/length of M_2, M_3, and M_4 (μm/μm)	29.64/0.8
2.	Width/length of load transistor M_1 (μm/μm)	0.8/1.702

3.2.3 Cascode Current Source

The cascode connection achieves a very high output resistance. Since this is a highly desirable characteristic for a current source, it is natural to explore the use of the cascode configuration for high-performance current sources. An NMOS cascode current source is shown in Fig.3.2.3. The bodies of all the transistors are connected to V_{SS}, and, therefore, the body effect of all the transistors is considered in the analysis.

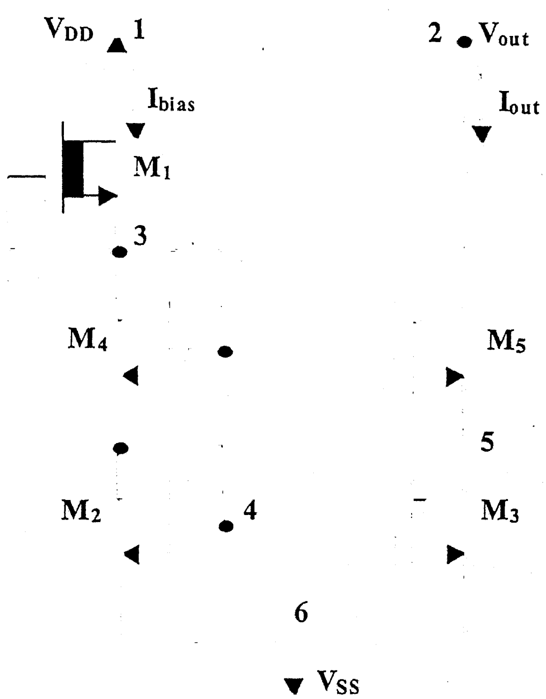

Fig. 3.2.3: Schematic of a cascode current source.

The transistors M_2, M_3, M_4, and M_5 are taken to be of the same size. Hence, there are four design variables, such as, widths and lengths of M_1 and M_2. Based on this assumption, a

training database for the cascode current source is created using SPICE simulation. This database is used to train both the RBF networks.

Based on the above training data, a module has been developed for the cascode current source, which gives the design variables as output from a given set of performance specifications as input to the routine. The output results along with the results obtained from SPICE simulation, and the design parameters obtained for a given set of performance specifications are shown in Tables 3.2.3.1 and 3.2.3.2 respectively. The results obtained from our design match well with those obtained from SPICE simulation.

Table 3.2.3.1

The design results obtained for a cascode current source and their comparison with those obtained from SPICE simulation.

Sl. No.	**Performance**	**Type**	**Specified value**	**Value obtained from program**	**Value obtained from SPICE**
1.	I_{out} (μA)	=	25	24.4	24.84
2.	R_{out} (MΩ)	≥	100	180	224.08
3.	V_{outmin} (V)	≤	1.5	1.396	1.467
4.	$AREA$ (μ^2)	≤	300	106.88	-

Table 3.2.3.2

The design variables obtained from the optimization program for a cascode current source.

Sl. No.	**Design variable**	**Value**
1.	Width/length of M_2, M_3, M_4, and M_5 (μm/μm)	11.595/0.8
2.	Width/length of load transistor M_1 (μm/μm)	5.365/13.005

3.3 THE COMMON SOURCE AMPLIFIER AS A GAIN STAGE

Single stage amplifiers are used virtually in every op-amp design. By using an active load, a significant amount of chip area can be saved. At the same time, since an active load presents a higher value of the output resistance, the net result is a much higher gain. The schematic of a CMOS common-source amplifier (NMOS driver with PMOS load) used as a gain stage is shown in Fig.3.3.1.

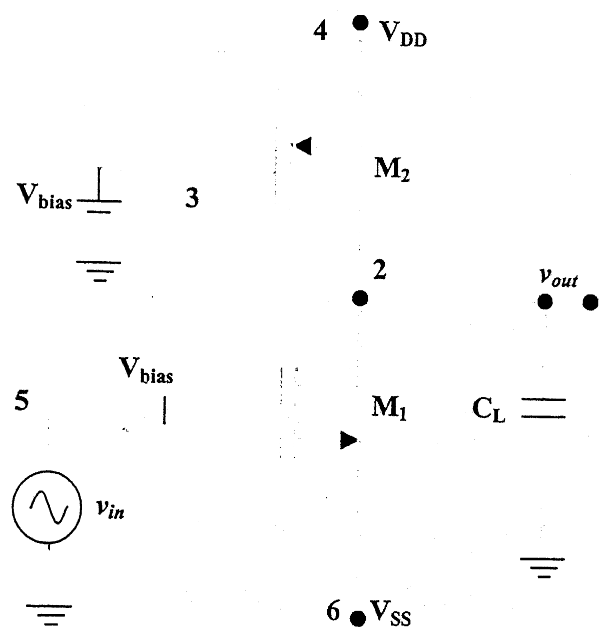

Fig. 3.3.1: Schematic of a CMOS common-source amplifier used as a gain stage.

This is an important block in CMOS technology, and is generally used as the second stage in a multi-stage op-amp configuration in order to increase the overall gain. A list of design objectives and the constraints for which the above circuit topology is optimized are given in Table 3.3.1. The design variables are the widths (W) and the lengths (L) of the transistors M_1 and M_2, and the dc bias voltage V_{bias}. Out of these five design variables, W_1, W_2, and V_{bias} are considered as independent design variables. The supply voltages V_{DD} and V_{SS} and the load capacitance (C_L) are taken as constant inputs ($\pm$ 5 V and 5 pF respectively).

Table 3.3.1

The list of performance specifications used for a CMOS common-source amplifier.

Sl. No.	**Symbol**	**Specification**	**Typical objective**
1.	*Gain*	Voltage gain	Maximize
2.	*UGF*	Unity-gain frequency	Maximize
3.	*SR*	Slew rate	Maximize
4.	*PD*	Power dissipation	Minimize
5.	*AREA*	Area	Minimize

The results obtained are compared with those obtained from SPICE simulation and are shown in Table 3.3.2. The obtained designed variables are given in Table 3.3.3. The variation of the voltage gain with the frequency for this circuit, as obtained from SPICE simulation, is plotted in Fig.3.3.2. The gain and the unity-gain frequency of the amplifier are read from the plot and are given in Table 3.3.2.

Table 3.3.2

The output results obtained for a common-source amplifier along with the results obtained from SPICE simulation.

Sl. No.	Performance	Type	Specified value	Value obtained from program	Value obtained from SPICE
1.	*Gain* (mag.)	≥	50	58.07	97.33
2.	*UGF* (MHz)	≥	20	45.61	27.49
3.	*SR* (V/μs)	≥	15	44.09	28.02
4.	*PD* (mW)	≤	5	2.3	1.09
5.	*AREA* (μ^2)	≤	130	129.94	-

Table 3.3.3

The design variables obtained from the optimization program for a common-source amplifier.

Sl. No.	Design variable	Value
1.	Width/length of M_1 (μm/μm)	66.61/0.8
2.	Width/length of M_2 (μm/μm)	95.81/0.8
3.	V_{bias} (V)	4.05

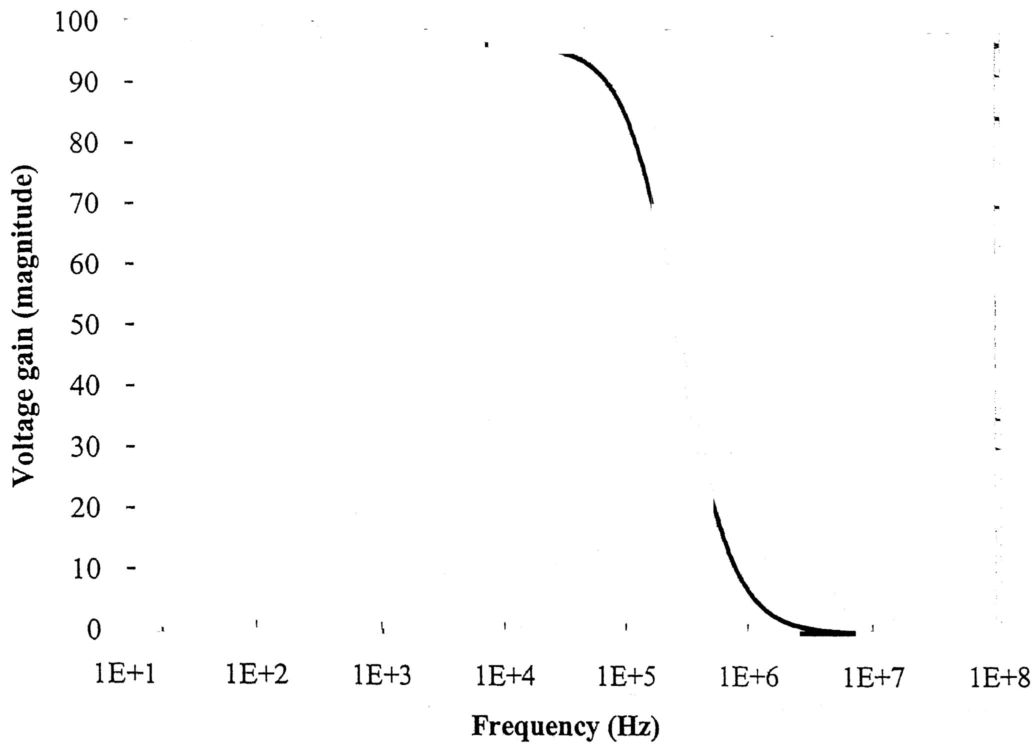

Fig. 3.3.2: The gain versus frequency plot for the common-source amplifier, as obtained from the SPICE simulation.

In this chapter, the synthesis procedure for some simple basic MOS building blocks, e.g., current sources (simple, Wilson, cascode) and the common-source amplifier as a gain stage are developed. Similarly, a variety of simple MOS circuits can be designed using the approach adopted in this work. The results obtained from our design showed a very close match with the results obtained for these circuits from actual SPICE simulations. Encouraged by these results, we extended our synthesis procedure to one of the most complex analog building blocks, i.e., the op-amp, and the results for this block are presented in the next chapter.

CHAPTER 4

DESIGN OF OPERATIONAL AMPLIFIERS

4.1 INTRODUCTION

The analog building block that is most often used is the operational amplifier. It is at the heart of many interface circuits, in particular, A/D and D/A converters, and switched-capacitor filters. An efficient and optimal design of op-amps is thus a corner stone of a design environment for many applications. The op-amp specifications for different applications vary so widely that it is impractical to store op-amps as library cells for all such applications. If the performance of the op-amp falls below a certain "threshold", the quality of the overall system will suffer. Hence, designing a good op-amp is a rather complicated multifaceted task [6]. In this chapter, the design procedures of two generic types of op-amps are discussed. These are the simple operational transconductance amplifier (OTA) and the basic three-stage op-amp. These op-amps are also known as general-purpose op-amps. The synthesis routines for each of the above mentioned op-amp topologies are described in respective subsections. The parameters of the circuit (i.e., the design variables), the output results obtained from the design routine for a given set of performance specifications, and their comparison with SPICE simulation are also presented.

The set of performance specifications and constraints for which the automation program has been developed in this work in order to design both the basic types of op-amp topologies are listed in Table 4.1.1.

Table 4.1.1

The set of performance specifications for which both types of op-amps are optimized in this work.

Sl. No.	Symbol	Specification	Typical objective
1.	*Gain*	Voltage gain of the amplifier	Maximize
2.	*UGF*	Unity-gain frequency	Maximize
3.	*PM*	Phase margin	Maximize
4.	*CMRR*	Common-mode rejection ratio	Maximize
5.	*SR*	Slew rate	Maximize
6.	*PD*	Power dissipation	Minimize
7.	*AREA*	Area	Minimize

4.2 THE SIMPLE OPERATIONAL TRANSCONDUCTANCE AMPLIFIER

The schematic of the simple operational transconductance amplifier (OTA), which is optimized in this work, is shown in Fig.4.2.1. This is one of the simplest CMOS op-amps having moderate gain, and is widely used in a variety of applications; one of them is driving on-chip loads, where minimum area is one of the most desirable aspects.

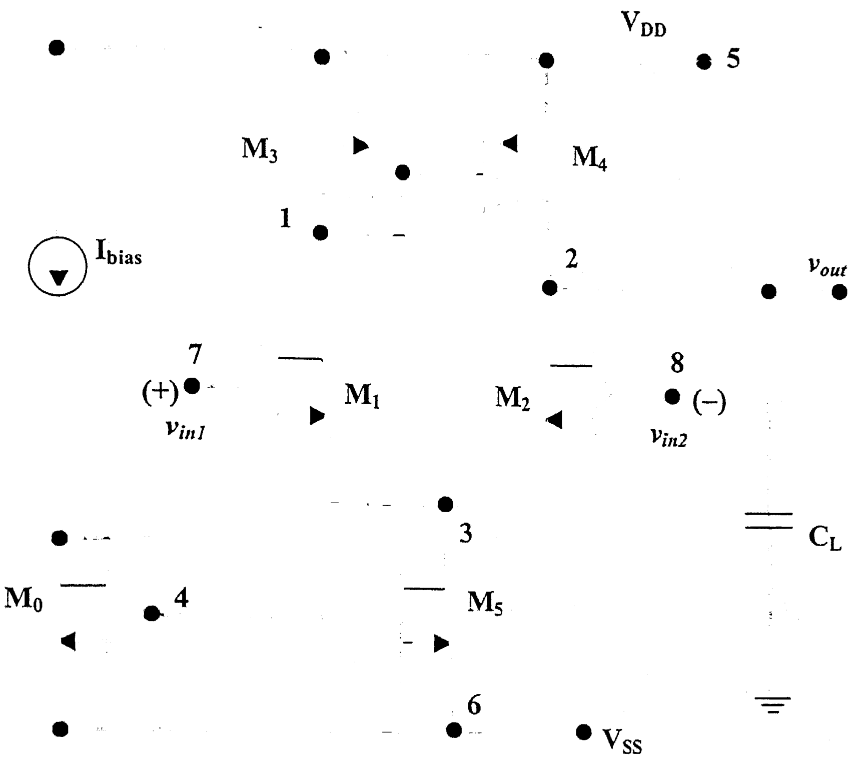

Fig. 4.2.1: The schematic representation of a simple operational transconductance amplifier (OTA).

The design variables for the simple OTA are the lengths (L) and the widths (W) of the individual transistors, and the magnitude of the bias current source I_{bias}. The supply voltages V_{DD} and V_{SS}, and the load capacitance (C_L) are taken as constants (± 5 V and 10 pF respectively). In order to obtain the independent design variables, which are to be varied during the optimization process, the following simplifications are made.

1. The transistors M_1 and M_2 are matched differential pair, and, therefore, $(W/L)_1 = (W/L)_2$.

2. The transistors M_3 and M_4 forming the current source also act as the active load for M_1/M_2, and, it is assumed that $(W/L)_3 = (W/L)_4$.
3. The bias current source consists of transistors M_5 and M_0, and, therefore, $(W/L)_5 = (W/L)_0$.

Hence, the number of independent design variables are reduced to four, i.e., the widths of the transistors M_0, M_1, and M_3, (i.e., W_0, W_1, and W_3), and the bias current I_{bias}. The channel lengths of all the transistors are taken to be equal to the minimum feature size, i.e., 0.8 μm. The upper and the lower bounds for the design parameter values are set to further restrict the search space. The bias current has been limited in the range from 10 μA to 40 μA. Using these independent design variables, a training database is created for the simple OTA, and, then, both the radial-basis function networks are trained and stored. With these networks, a module is created to design the OTA. In the module, the penalty function method is used as a constrained optimization routine. This method is a single objective, multiple constraint optimization process. In order to design this architecture, in the optimization routine, the power dissipation is taken as the objective function [refer to Eqn.(2.1)], which is to be minimized, and all other specifications are taken as constraints. The performances of the circuit are computed using the reverse RBF network (described as RBF2 in Chapter 2).

The results obtained are compared with those obtained from SPICE simulation and are shown in Table 4.2.1. The obtained designed variables are given in Table 4.2.2. The variations of the voltage gain and the phase with the frequency, as obtained from SPICE simulation for the designed topology is plotted in Fig.4.2.2. The values obtained from SPICE simulation are in accordance with those predicted by our optimizer.

Table 4.2.1

The results obtained from the design of simple OTA for a particular set of performance specifications along with those obtained from SPICE simulation.

Sl. No.	Performance	Specified value	Value obtained from program	Value obtained from SPICE
1.	*Gain*	100	128.342	100.5
2.	*UGF* (MHz)	40	45.326	41.73
3.	*PM* (degree)	85	90.758	89.83
4.	*CMRR* (dB)	65	73.998	67.63
5.	*SR* (V/μs)	3	5.535	5.349
6.	*PD* (mW)	2	1.648	1.516
7.	*AREA* (μ^2)	300	295.547	–

Table 4.2.2

The set of design variables obtained from the design of the simple OTA for a set of performance specifications.

Sl. No.	Design variable	Value
1.	Width/length of M_1, M_2 (μm/μm)	78.942/0.8
2.	Width/length of M_3, M_4 (μm/μm)	44.496/0.8
3.	Width/length of M_0, M_5 (μm/μm)	61.279/0.8
4.	I_{bias} (μA)	30.601

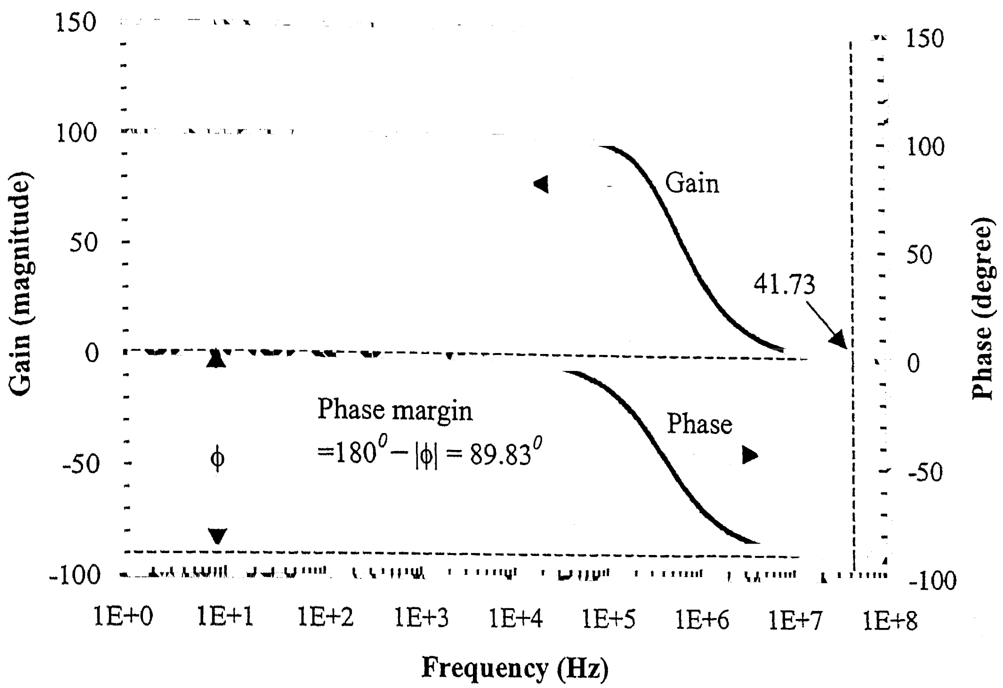

Fig. 4.2.2: The gain and the phase versus frequency plots for the simple OTA, as obtained from the SPICE simulation. The unity-gain frequency and the phase margin are also shown in the figure.

4.3 THE COMPENSATED THREE-STAGE OPERATIONAL TRANSCONDUCTANCE AMPLIFIER

The schematic of a basic three-stage CMOS op-amp is shown in Fig.4.3.1. It consists of three stages, the first of which is a differential stage with NMOS input devices M_1 and M_2, and the PMOS current mirror M_3 and M_4, also acting as the active load for M_1/M_2. The second stage is a simple CMOS gain stage with M_6 as the driver and M_7 as the active load.

The output of the gain stage is connected to its input through the compensating capacitor C_C. The third stage is a voltage follower output stage consisting of NMOS devices M_8 and M_9.

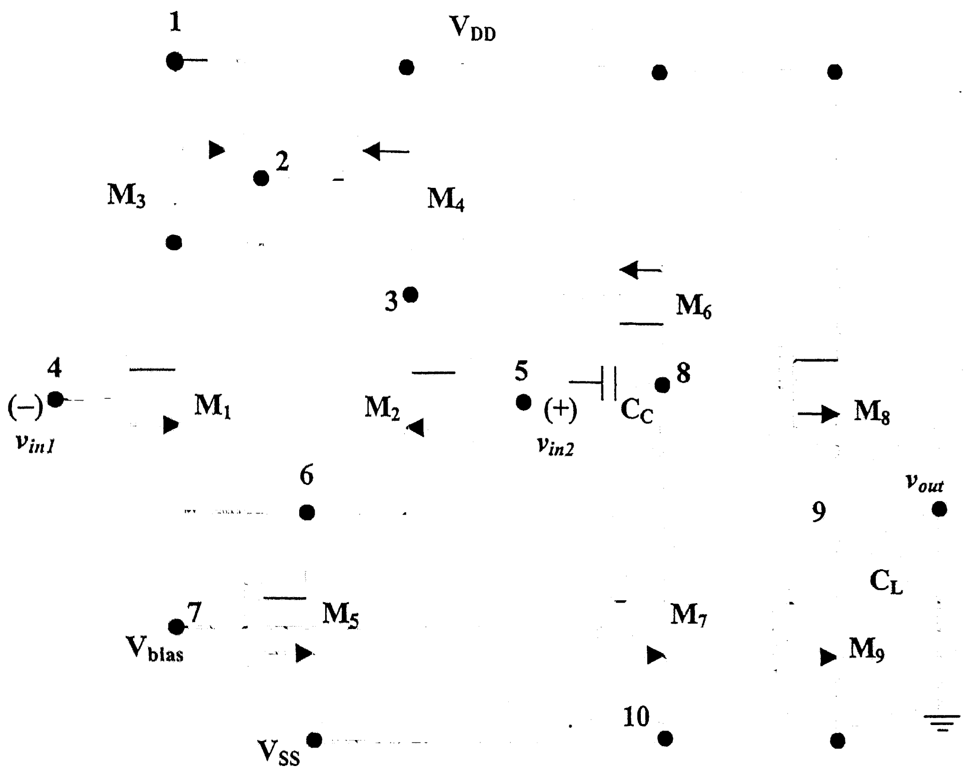

Fig. 4.3.1: Schematic of the compensated three-stage CMOS operational transconductance amplifier (OTA).

The design variables for this architecture are the lengths (L) and the widths (W) of the individual transistors, the magnitude of the bias voltage V_{bias}, and the value of the compensation capacitor C_C. The supply voltages V_{DD} and V_{SS}, and the load capacitance (C_L) are taken as constants ($\pm$ 5 V and 10 pF respectively). In order to get the independent design variables, which are to be varied during the optimization process, the following simplifications are made.

1. The input transistors M_1 and M_2 are matched differential pair, and, therefore, $(W/L)_1 = (W/L)_2$.
2. The transistors M_3 and M_4 are current sources also acting as the active load for M_1/M_2, and, it is assumed that $(W/L)_3 = (W/L)_4$.
3. The transistors M_8 and M_9 are used as output stage and, in order to get symmetrical rise and fall times it is assumed that $(W/L)_8 = (W/L)_9$.

Hence, eight independent design variables are obtained for this topology, i.e., the widths of the transistors M_1, M_3, M_5, M_6, M_7, and M_8, (i.e., W_1, W_3, W_5, W_6, W_7, and W_8), the bias voltage V_{bias}, and the value of the compensation capacitance C_C. The channel lengths of all the transistors are taken to be equal to the minimum feature size, i.e., 0.8 μm. The bias voltage and the compensation capacitor values range from 3.5 V to 4.1 V and 5 pF to 20 pF respectively, since, in general, the values outside this range are not suitable for practical applications. Using these independent design variables, a training database is created for this op-amp architecture, and, then, both the radial-basis function networks are trained and stored. With these networks, a module is created to design this circuit. Similar to the design of the simple OTA, in order to design this architecture, the power dissipation is taken as the objective function [refer to Eqn.(2.1)], which is to be minimized, and all other specifications are taken as constraints in the optimization routine.

The obtained results are compared with those obtained from SPICE simulation and are shown in Table 4.3.1. The designed variables obtained from our program are given in Table 4.3.2. The variations of the voltage gain with the frequency and the phase with the

frequency, as obtained from SPICE simulation for the designed topology, are plotted in Figs.4.3.2 and 4.3.3 respectively.

Table 4.3.1

The results obtained from the design of the basic three-stage op-amp for a particular set of performance specifications along with those obtained from SPICE simulation.

Sl. No.	Performance	Specified value	Value obtained from program	Value obtained from SPICE
1.	*Gain* (dB)	50	63.14	55.32
2.	*UGF* (MHz)	2	2.09	2.26
3.	*PM* (degree)	40	72.62	40.08
4.	*CMRR* (dB)	60	64.61	65.62
5.	*SR* (V/μs)	2	15.2	2.58
6.	*PD* (mW)	50	32.32	15.05
7.	*AREA* (μ^2)	200	200.17	–

From Table 4.3.1, it can be seen that some performance specifications (e.g., PM and SR), as obtained from our program, are not matched accurately with the SPICE simulation. In the program, these specifications are obtained from the design variables using RBF2. Now, the values of SR and PM depend on the bias voltage V_{bias} and the compensation capacitance C_C. It can be easily verified using SPICE simulations that this dependence is highly nonlinear. The RBF2, which works on the global nonlinearity, i.e., the nonlinearity created by all the design variables, is unable to track this local nonlinearity. Hence, it gives a bit inaccurate results in this region.

Table 4.3.2

The set of design variables obtained from the design of the basic three-stage op-amp for a set of performance specifications.

Sl. No.	Design variable	Value
1.	Width/length of M_1, M_2 (μm/μm)	0.8/0.8
2.	Width/length of M_3, M_4 (μm/μm)	94.24/0.8
3.	Width/length of M_5 (μm/μm)	0.8/0.8
4.	Width/length of M_6 (μm/μm)	56.93/0.8
5.	Width/length of M_7 (μm/μm)	0.8/0.8
6.	Width/length of M_8, M_9 (μm/μm)	0.8/0.8
7.	V_{bias} (V)	–3.815
8.	Compensating capacitor C_C (pF)	11.07

From Table 4.3.2, it can be noted that the output stage consisting of M_8/M_9, has an aspect ratio of unity. Since they drive current to the load, hence, their physical sizes should be the largest. Large aspect ratio is needed to increase the slew rate, but this also leads to an increase in the active area and the power dissipation. From the results, we can see that the specified slew rate is achieved and other constraints are also met with the output stage having an aspect ratio of unity. From Appendix-I, it can be seen that initially (i.e., while providing inputs to the optimization routine), the aspect ratio of that stage was taken to be equal to 125 (the largest aspect ratio we can have), but finally, the optimizer changed the aspect ratio to unity according to the specifications.

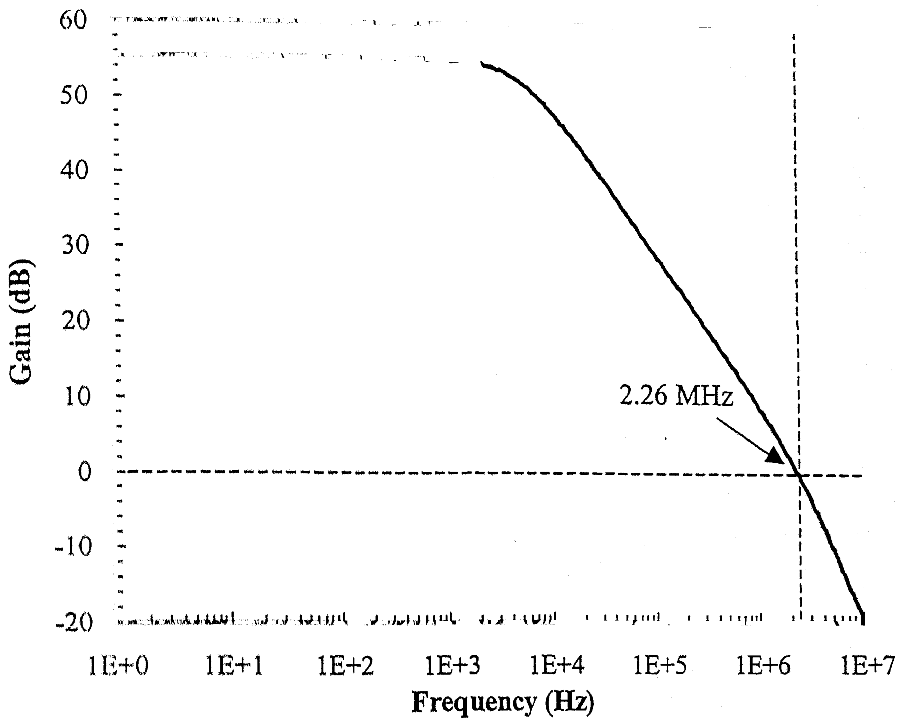

Fig. 4.3.2: The gain versus frequency plot for the basic three-stage op-amp, as obtained from the SPICE simulation. The unity-gain frequency is also shown in the figure.

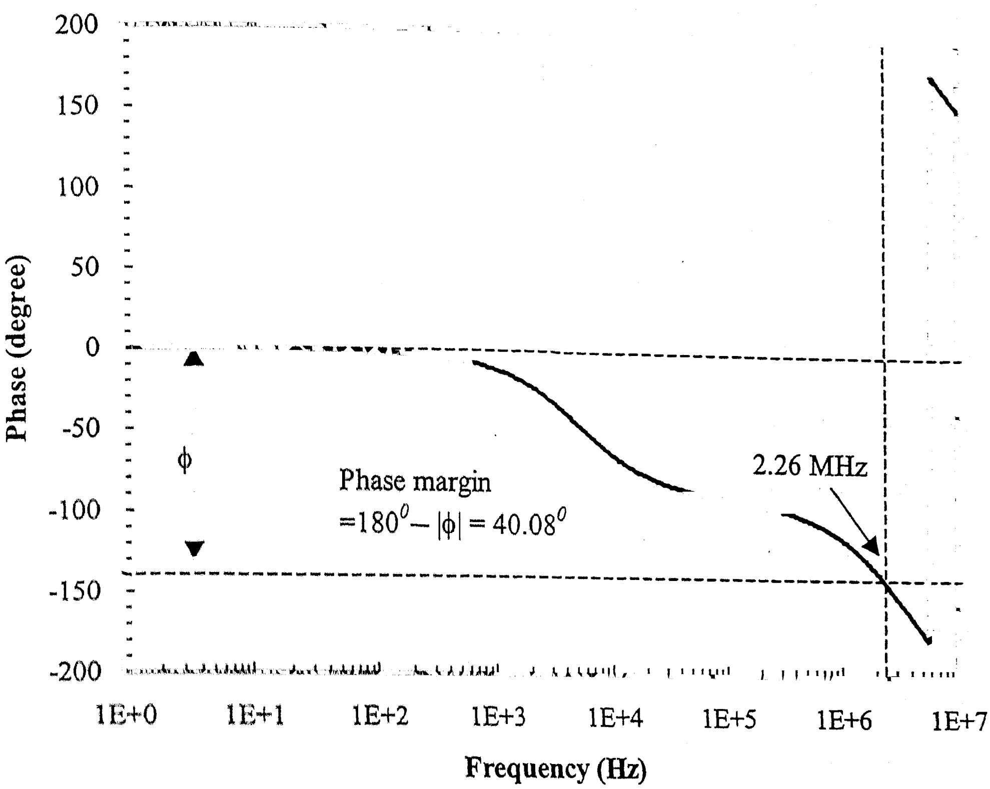

Fig. 4.3.3: The phase versus frequency plot for the basic three-stage op-amp, as obtained from the SPICE simulation. The phase margin is also shown in the figure.

The compensation capacitance C_C causes a positive zero, i.e., a zero that is located in the right-hand complex plane. At high frequencies, feedforward through C_C tends to overwhelm the normal gain path via the transconductance g_m of the second stage. The feedforward path does not have the 180^0 phase shift of the normal gain stage, and thus the gain path loses an inverting stage. This phenomenon can be noted from Fig.4.3.3, where the phase becomes positive at high frequencies.

Two effective means have evolved for eliminating the effect of the right half-plane zero. One approach is to insert a source follower in series with the compensation capacitance C_C. The source follower takes input from the output of the second stage (v_o) and produces output back through the compensation capacitor to the input of the second stage (v_i) as shown in Fig.4.3.4. Its gain can be less than unity and is denoted by A_C. This buffer stage acts as a unidirectional voltage follower, which prevents the propagation of the signals forward through the capacitor. The inverter stage charges the output node and the feedback through the compensation capacitance is also obtained. This works well, although it requires more devices and dc bias current to construct the source follower [24].

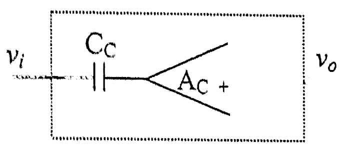

Fig. 4.3.4: The source follower in the path from the output of the second stage back through the compensation capacitor C_C [24].

An even simpler approach is to insert a nulling resistor R_C in series with the compensation capacitor C_C. Then, the location of the zero can be calculated as [20]

$$z = \frac{1}{C_C\left(\frac{1}{g_m} - R_C\right)}. \tag{4.1}$$

Equation (4.1) shows that the zero vanishes when R_C is made equal to $1/g_m$ of the second stage. In fact, the value of the resistor R_C can be further increased to move the zero into the left half-plane in order to improve the phase margin of the op-amp.

CHAPTER 5

SUMMARY AND CONCLUSION

In this work, we have presented a new methodology for automating the design of analog integrated circuits. The analog DA has become a potential research area because of the pervasive trend in recent years towards the integration of whole systems into single chip VLSICs, which requires both digital functional units and dedicated analog interface subsystems (such as A/D converters and filters) to be incorporated on the same chip. Many digital parts of such chips can nowadays be synthesized rapidly and reliably using CAD tools developed for semi-custom design methods such as gate arrays, standard cells, and macro cells. On the other hand, analog interface subsystems still need to be entirely handcrafted by a specialist. Therefore, the design time and the cost associated with dedicated analog interface components often constitute a bottleneck in semi-custom design of mixed analog/digital systems.

In our work, we have parametrically optimized the values of the design variables for a given set of input specifications (e.g., for an op-amp, these could be the open loop gain, unity gain bandwidth, area, etc.), where the topology of the circuit is assumed to be known apriori. Two radial basis function (RBF) networks are used in this work: a forward RBF network (RBF1) is used to generate the initial values of the design variables from the given performance specifications and a reverse RBF network (RBF2) is used to evaluate the circuit performances from the design variables. The design of a neural network can be viewed as a

curve-fitting problem, where the number of independent parameters is large. The construction of an RBF network involves three entirely different layers. The input layer is made up of the source nodes, the second layer is a hidden layer of high enough dimension, and the output layer supplies the response of the network to the activation patterns applied to the input layer. The transformation from the input space to the hidden-unit space is nonlinear, whereas the transformation from the hidden-unit space to the output space is linear. The Gaussian function is used in the mapping of the input layer to the hidden layer. A database is created for each circuit using SPICE, in order to train both the RBF networks. With the help of this database, the weights of the RBF networks are updated and the information is stored in a file for future use.

The initial solution, as obtained from RBF1, is the starting point for the optimization routine, which uses the penalty function method, which is a constrained optimization technique. In this method, the constrained problem is transformed into a sequence of unconstrained problems by adding penalty terms to the objective function, the amount of which depends on the extent of the constraint violation for each constraint. It works in a series of sequences, each time modifying a set of penalty parameters and starting a sequence with the solution obtained in the previous sequence. This optimization routine uses RBF2 to evaluate the circuit performances from the design variables. Then, the performance specifications and the design variables as obtained from the optimization routine are stored in an output file. These design variables can be used in SPICE simulation to verify the circuit performances with the given performance specifications.

FUTURE WORK

The following modifications can be incorporated in our work in order to make it a complete design tool, which can be used more easily by the designers.

- The layout of the designed circuit has to be done to extract the parasitic capacitances of the interconnects and then optimize the circuit accordingly.
- Different optimization algorithms (e.g., genetic algorithm, simulated annealing, etc.), which give global optimum solutions, can be used to get a better optimized solution.
- A completely different way in which this work could be expanded would be to support larger circuits (such as A/D converter, phase locked loop, etc.) via hierarchy and macromodeling.
- This work is carried out assuming fixed topology for a circuit. It could be combined with a topology selection method in order to make it more complete.
- A graphical user interface (GUI) could be developed to facilitate the use of the tool for the novice designers.

REFERENCES

1. E.S. Ochotta, R.A. Rutenbar, and L.R. Carley, "Synthesis of high-performance analog circuits in ASTRX/OBLX", *IEEE Trans. Computer-Aided Design,* vol. 15, no. 3, pp. 273-293, March 1996.
2. M. Ismail and J. Franca, *Introduction to Analog VLSI Design Automation,* Kluwer Academic Publishers, London, 1990.
3. C. Toumazou and C.A. Makris, "Analog IC design automation: Part I – Automated circuit generation: New concepts and methods", *IEEE Trans. Computer-Aided Design,* vol. 14, no. 2, pp. 218-238, February 1995.
4. L.R. Carley and R.A. Rutenbar, "How to automate analog IC design", *IEEE Spectrum,* vol. 25, no. 8, pp. 26-30, August 1988.
5. G. Kelson, "Design automation techniques for analog VLSI", *VLSI Design,* vol. 5, no. 1, pp. 78-82, January 1985.
6. W. Nye, D.C. Riley, A. Sangiovanni, and A.L. Tits, "DELIGHT.SPICE: An optimization-based system for the design of integrated circuits", *IEEE Trans. Computer-Aided Design,* vol. 7, no. 4, pp. 501-519, April 1998.
7. J.M. Shyu and A. Sangiovanni-Vincetelli, "ECSTASY: A new environment for IC design optimization", *Proc. IEEE Inter. Conf. Computer-Aided Design (ICCAD),* pp. 484-487, November 1988.
8. J.C. Lai, J.S. Kueng, H.J. Chen, and F.J. Fernandez, "ADOPT – A CAD tool for analog circuit design", *IEEE Circuits Devices Magazine (M-C&D),* vol. 4, no. 2, pp. 29-30, March 1988.

9. R. Phelps, M. Krasniciki, R.A. Rutenbar, L.R. Carley, and J.R. Hellums, "Anaconda: Simulation-Based synthesis of analog circuits via stochastic pattern search", *IEEE Trans. Computer-Aided Design*, vol. 19, no. 6, pp. 703-717, June 2000.

10. F. El-Turkey and E.E. Perry, "BLADES: An artificial intelligence approach to analog circuit design", *IEEE Trans. Computer-Aided Design*, vol. 8, no. 12, pp. 1247-1265, December 1989.

11. R. Harijani, R.A. Rutenbar, and L.R. Carley, "OASYS: A framework for analog circuit synthesis", *IEEE Trans. Computer-Aided Design*, vol. 8, no. 6, pp. 680-691, June 1990.

12. E. Berkcan, M. Abreu, and W. Laughton, "Analog compilation based on successive decompositions", *Proc. ACM/IEEE Design Automat. Conf. (DAC)*, pp. 369-375, June 1988.

13. M.R. Degrauwe, O. Nys, and H.J. Oguey, "IDAC: An interactive design tool for analog CMOS circuits", *IEEE J. Solid-State Circuits*, vol. SC-22, no. 6, pp. 1106-1116, December 1987.

14. H.Y. Koh, C H. Sequin, and P.R. Grey, "OPASYN: A compiler for CMOS operational amplifiers", *IEEE Trans. Computer-Aided Design*, vol. 9, no. 2, pp. 113-125, February 1990.

15. H. Odonera, H. Kanbara, and K. Tamaru, "Operational-amplifier compilation with performance optimization", *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 466-473, April 1990.

16. G.G.E. Gielen, H.C.C. Walscharts, and W.M. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing", *IEEE J. Solid-State Circuits*, vol. SC-25, no. 3, pp. 707-713, June 1990.

17. K. Deb, *Optimization for Engineering Design,* Prentice Hall India (Pvt.) Ltd., New Delhi, 1995.

18. S.S. Haykin, *Foundations of Neural Networks,* Prentice Hall, New Jersey, 1994.

19. S. Chen, C.F.N. Cowan, and P.M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks", *IEEE Trans. Neural Networks,* vol. 2, no. 2, pp. 302-309, March 1991.

20. P.R. Gray and R.G. Meyer, *Analysis and Design of Analog Integrated Circuits,* 3rd Edition, John Wiley and Sons, New York, 1993.

21. R. Gregorian and G.C. Temes, *Analog MOS Integrated Circuits for Signal Processing,* Wiley and Sons, New York, 1984.

22. R.J. Baker, H.L. Wi, and D.E. Boyce, *CMOS: Circuit design, layout and simulation,* IEEE Press, The IEEE Inc., New York, 1998.

23. J.A. Anderson, *An introduction to Neural networks,* Eastern Economy Edition, Prentice Hall, India, 1998.

24. K.R. Laker, W.M.C. Sansen, *Design of Analog Integrated Circuits and Systems,* McGraw-Hill, New Jersey, 1994.

25. C. Toumazou and C.A. Makris, "Analog IC design automation: Part II – Automated circuit correction by qualitative reasoning", *IEEE Trans. Computer-Aided Design,* vol. 14, no. 2, pp. 239-254, February 1995.

26. B. Sahu, "A fuzzy logic based approach for parametric optimization of analog circuits", *M. Tech. Thesis,* IIT Kanpur, December 1999.

A CASE STUDY

In this work, we briefly introduce the sequence of operations that underlies our new formulation for analog synthesis. Here, we present a complete design example in order to make concrete the entire design path, starting from the problem and ending at the solution. The example explained in this section involves sizing and biasing the basic three-stage compensated operational amplifier as shown in Fig.4.3.1 in order to meet the specifications given in Table A1.1.

Table A1.1

The performance specifications for the basic three-stage compensated op-amp, as shown in Fig.4.3.1.

Sl. No.	Performance Specification	Specified value
1.	*Gain* (dB)	50
2.	*UGF* (MHz)	2
3.	*PM* (degree)	40
4.	*CMRR* (dB)	60
5.	*SR* (V/μs)	2
6.	*PD* (mW)	50
7.	*AREA* (μ^2)	200

At first, the radial basis function networks should be trained; hence, a training database is created using SPICE. Sample values of the training data are shown in the attached computer output and also, the mean for each type of input data is calculated and shown.

THE MOS TECHNOLOGY FILE

Table A2.1

A set of process parameters for a typical silicon-gate *n*-well CMOS process with 0.8 μm minimum feature size, which is used in the design of all MOS circuit topologies in this work (taken from [20]).

Sl. No.	Parameter	Symbol	Value	
			n-channel transistor	p-channel transistor
1.	Substrate doping (atoms/cm^3)	N_A, N_D	4×10^{15}	3×10^{16}
2.	Gate oxide thickness (nm)	t_{ox}	15	15
3.	Channel mobility (cm^2/V-sec)	μ_n, μ_p	550	250
4.	Minimum drawn channel length (μm)	L_{drawn}	0.8	0.8
5.	Source, drain junction depth (μm)	X_j	0.2	0.3
6.	Source, drain side diffusion (μm)	L_d	0.12	0.18
7.	Overlap capacitance per unit gate width (fF/μm)	C_{ol}	0.12	0.18
8.	Source/drain-bulk junction capacitance per unit source/drain area (fF/μm^2) (zero bias)	C_{j0}	0.18	0.3
9.	Source/drain bulk junction capacitance exponent	n	0.5	0.5

10.	Source/drain-periphery capacitance per unit source/drain periphery (fF/µm) (zero bias)	C_{jsw0}	1.0	2.2
11.	Source/drain-periphery capacitance exponent	n	0.5	0.5
12.	Nominal threshold voltage (V)	V_t	0.7	−0.7
13.	Channel length modulation parameter (V^{-1})	λ	0.1	0.05
14.	Poly gate sheet resistance (Ω/□)	R_S	10	10
15.	Surface-state density (cm^{-2})	N_{ss}	10^{11}	10^{11}